CLAIMS

1. A method for creating a cache hierarchy in a computer system, the method comprising an act of:

(A) creating a software cache hierarchy having at least two software caches that are interrelated to form the cache hierarchy, the at least two software caches including at least a first software cache and a second software cache, wherein the first and second software caches employ different hashing techniques for mapping an address into the first and second software caches.

2. The method of claim 1, wherein the act (A) comprises an act of creating the software cache hierarchy so that each of the first and second software caches is a set associative cache.

3. The method of claim 1, wherein the act (A) comprises an act of arranging the first and second software caches in a stacked hierarchy, such that addresses are compared first against the first cache, and are compared against the second cache in response to a miss in the first cache.

4. The method of claim 3, wherein the act (A) comprises an act of arranging the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is demoted to the second cache, and data that hits in the second cache is promoted to the first cache.

5. The method of claim 3, wherein the act (A) comprises an act of arranging the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is automatically demoted to the second cache, and data that hits in the second cache is automatically promoted to the first cache, wherein the automatic promotion and demotion of data is automatically performed by the cache hierarchy.

6. The method of claim 1, further comprising an act of dynamically reconfiguring the cache hierarchy without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

7. The method of claim 4, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy, and wherein the act (A) comprises an act of creating an overflow area in the cache hierarchy, beneath the bottom cache, to which data replaced out of the bottom cache can be stored so that data replaced out of the bottom cache need not be stored outside of the cache hierarchy prior to allowing new data to be written in the bottom cache.

8. The method of claim 7, wherein the act (A) comprises an act of creating the overflow area so that in response to an access request to the cache hierarchy, the overflow area is not examined to determine whether the access request matches data in the overflow area.

9. The method of claim 1, wherein the at least two software caches comprise a number of software caches, and wherein the method further comprises an act (B) of modifying the number of software caches in the cache hierarchy.

10. The method of claim 9, wherein the act (B) is performed dynamically without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

11. The method of claim 1, further comprising an act (B) of altering a storage medium used to implement at least one of the caches in the cache hierarchy.

12. The method of claim 11, wherein the act (B) is performed dynamically without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

13. The method of claim 1, wherein the act (A) comprises an act of arranging the first and second software caches in a stacked hierarchy, such that addresses are

compared first against the first cache, and are compared against the second cache only in response to a miss in the first cache.

14. The method of claim 3, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy;

wherein the act (A) comprises an act of arranging the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is demoted to the second cache; and

wherein the act (A) comprises an act of creating an overflow area in the cache hierarchy, beneath the bottom cache, to which data replaced out of the bottom cache can be stored so that data replaced out of the bottom cache need not be stored outside of the cache hierarchy prior to allowing new data to be written in the bottom cache.

15. A computer for use in a computer system, the computer comprising:

at least one processor programmed to implement a software cache hierarchy having at least two software caches that are interrelated to form the cache hierarchy, the at least two software caches including at least a first software cache and a second software cache, wherein the first and second software caches employ different hashing techniques for mapping an address into the first and second software caches.

16. The computer of claim 15, wherein the at least one processor is programmed to create the software cache hierarchy so that each of the first and second software caches is a set associative cache.

17. The computer of claim 15, wherein the at least one processor is programmed to arrange the first and second software caches in a stacked hierarchy, such that addresses are compared first against the first cache, and are compared against the second cache in response to a miss in the first cache.

18. The computer of claim 17, wherein the at least one processor is programmed to arrange the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is demoted to the second cache, and data that hits in the second cache is promoted to the first cache.

19. The computer of claim 17, wherein the at least one processor is programmed to arrange the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is automatically demoted to the second cache, and data that hits in the second cache is automatically promoted to the first cache, wherein the automatic promotion and demotion of data is automatically performed by the cache hierarchy.

20. The computer of claim 15, where the at least one processor is programmed to dynamically reconfigure the cache hierarchy without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

21. The computer of claim 18, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy, and wherein the at least one processor is programmed to create an overflow area in the cache hierarchy, beneath the bottom cache, to which data replaced out of the bottom cache can be stored so that data replaced out of the bottom cache need not be stored outside of the cache hierarchy prior to allowing new data to be written in the bottom cache.

22. The computer of claim 21, wherein the at least one processor is programmed to create the overflow area so that in response to an access request to the cache hierarchy, the overflow area is not examined to determine whether the access request matches data in the overflow area.

23. The computer of claim 15, wherein the at least two software caches comprise a number of software caches, and wherein the at least one processor is programmed to modify the number of software caches in the cache hierarchy.

24. The computer of claim 23, wherein the at least one processor is programmed to modify the number of software caches in the cache hierarchy dynamically without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

25. The computer of claim 15, wherein the at least one processor is programmed to alter a storage medium used to implement at least one of the caches in the cache hierarchy.

26. The computer of claim 25, wherein the at least one processor is programmed to alter the storage medium used to implement at least one of the caches in the cache hierarchy dynamically, without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

27. The computer of claim 15, wherein the at least one processor is programmed to arrange the first and second software caches in a stacked hierarchy, such that addresses are compared first against the first cache, and are compared against the second cache only in response to a miss in the first cache.

28. The computer of claim 17, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy;

wherein the at least one processor is programmed to arrange the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is demoted to the second cache; and

wherein the at least one processor is programmed to create an overflow area in the cache hierarchy, beneath the bottom cache, to which data replaced out of the bottom cache can be stored so that data replaced out of the bottom cache need not be stored outside of the cache hierarchy prior to allowing new data to be written in the bottom cache.

29. A computer readable medium encoded with a program for execution on a computer system which, when executed, performs a method for creating a cache hierarchy, the method comprising an act of:

(A) creating a software cache hierarchy having at least two software caches that are interrelated to form the cache hierarchy, the at least two software caches including at least a first software cache and a second software cache, wherein the first and second software caches employ different hashing techniques for mapping an address into the first and second software caches.

30. The computer readable medium of claim 29, wherein the act (A) comprises an act of creating the software cache hierarchy so that each of the first and second software caches is a set associative cache.

31. The computer readable medium of claim 29, wherein the act (A) comprises an act of arranging the first and second software caches in a stacked hierarchy, such that addresses are compared first against the first cache, and are compared against the second cache in response to a miss in the first cache.

32. The computer readable medium of claim 31, wherein the act (A) comprises an act of arranging the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is demoted to the second cache, and data that hits in the second cache is promoted to the first cache.

33. The computer readable medium of claim 31, wherein the act (A) comprises an act of arranging the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is automatically demoted to the second cache, and data that hits in the second cache is automatically promoted to the first cache, wherein the automatic promotion and demotion of data is automatically performed by the cache hierarchy.

34. The computer readable medium of claim 29, wherein the method further comprises an act of dynamically reconfiguring the cache hierarchy without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

35. The computer readable medium of 32, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy, and wherein the act (A) comprises an act of creating an overflow area in the cache hierarchy, beneath the bottom cache, to which data replaced out of the bottom cache can be stored so that data replaced out of the bottom cache need not be stored outside of the cache hierarchy prior to allowing new data to be written in the bottom cache.

36. The computer readable medium of claim 35, wherein the act (A) comprises an act of creating the overflow area so that in response to an access request to the cache hierarchy, the overflow area is not examined to determine whether the access request matches data in the overflow area.

37. The computer readable medium of claim 29, wherein the at least two software caches comprise a number of software caches, and wherein the method further comprises an act (B) of modifying the number of software caches in the cache hierarchy.

38. The computer readable medium of claim 37, wherein the act (B) is performed dynamically without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

39. The computer readable medium of claim 29, wherein the method further comprises an act (B) of altering a storage medium used to implement at least one of the caches in the cache hierarchy.

40. The computer readable medium of claim 39, wherein the act (B) is performed dynamically without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

41. The computer readable medium of claim 29, wherein the act (A) comprises an act of arranging the first and second software caches in a stacked hierarchy, such that addresses are compared first against the first cache, and are compared against the second cache only in response to a miss in the first cache.

42. The computer readable medium of claim 31, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy;

wherein the act (A) comprises an act of arranging the first and second software caches in the stacked hierarchy so that data replaced out of the first cache is demoted to the second cache; and

wherein the act (A) comprises an act of creating an overflow area in the cache hierarchy, beneath the bottom cache, to which data replaced out of the bottom cache can be stored so that data replaced out of the bottom cache need not be stored outside of the cache hierarchy prior to allowing new data to be written in the bottom cache.

43. A method for determining whether an address hits in a cache hierarchy in a computer system, the cache hierarchy including at least two software caches that are interrelated to form the cache hierarchy, the at least two software caches including at least a first software cache and a second software cache, the method comprising acts of:

applying a first hashing algorithm to the address to map the address into the first software cache;

determining whether the address hits or misses in the first software cache; and

when it is determined that the address misses in the first software cache, performing the acts of:

applying a second hashing algorithm to the address to map the address into the second software cache, the second hashing algorithm being different from the first hashing algorithm; and

determining whether the address hits in the second software cache.

44. The method of claim 43, wherein each of the first and second software caches is a set associative cache.

45. The method of claim 43, wherein each cache has a plurality of entries each having associated data, and wherein the method further comprises acts of:

when the first cache is full and a new entry is to be added to the first cache, replacing an old entry out of the cache and demoting the old entry to the second cache; and

when it is determined that the address hits an entry in the second cache, promoting the entry to the first cache.

46. The method of claim 43, wherein each cache has a plurality of entries each having associated data, and wherein the method further comprises acts of:

when the first cache is full and a new entry is to be added to the first cache, replacing an old entry out of the cache and automatically demoting the old entry to the second cache; and

when it is determined that the address hits a hit entry in the second cache, automatically promoting the hit entry to the first cache, wherein the automatic promotion and demotion of cache entries is automatically performed by the cache hierarchy.

47. The method of claim 43, further comprising an act of dynamically reconfiguring the cache hierarchy without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

48. The method of claim 46, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy and an overflow area in the cache hierarchy, beneath the bottom cache, the method further comprising an act of:

storing data replaced out of the bottom cache to the overflow area so that data replaced out of the bottom cache need not be stored outside of the cache hierarchy prior to allowing new data to be written in the bottom cache.

49. The method of claim 48, further comprising an act of:

not examining the overflow area in response to an access request to the cache hierarchy to determine whether the access request matches data in the overflow area.

50. The method of claim 43, wherein the at least two software caches comprise a number of software caches, and wherein the method further comprises an act of modifying the number of software caches in the cache hierarchy.

51. The method of claim 50, wherein the act of modifying the number of software caches in the cache hierarchy is performed dynamically without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

52. The method of claim 43, further comprising an act of altering a storage medium used to implement at least one of the caches in the cache hierarchy.

53. The method of claim 52, wherein the act of altering a storage medium is performed dynamically without reconfiguring any application executing on the computer system that accesses the cache hierarchy.

54. The method of claim 43, wherein the stacked hierarchy comprises a plurality of caches including a bottom cache disposed at a bottom of the stacked hierarchy and an overflow area in the cache hierarchy, beneath the bottom cache, wherein each cache has a plurality of entries each having associated data, and wherein the method further comprises acts of:

when the first cache is full and a first new entry is to be added to the first cache, replacing a first old entry out of the first cache and demoting the first old entry to the second cache; and

when the bottom cache is full and a second new entry is to be added to the bottom cache, replacing a second old entry out of the bottom cache and storing the second old entry to the overflow area so that the second old entry need not be stored

outside of the cache hierarchy prior to allowing the second new entry to be written in the bottom cache.

55. A method for managing a cache arrangement in a computer system, the cache arrangement having a plurality of caches that are interrelated to form the cache arrangement, the method comprising an act of:

(A) dynamically reconfiguring the cache arrangement without reconfiguring an application, executing on the computer system, that accesses the cache arrangement.

56. The method of claim 55, wherein the plurality of caches comprises a plurality of software caches.

57. The method of claim 55, wherein at least one of the plurality of caches is a set associative cache comprising a number of cache groups each including a number of cache sets, and wherein the act (A) comprises an act of dynamically reconfiguring the at least one of the plurality of caches by modifying at least one of the number of cache groups and the number of cache sets.

58. The method of claim 55, wherein the plurality of caches are arranged in a stacked hierarchy, such that addresses are compared first against a first cache in the stacked hierarchy, and are compared against a second cache in the stacked hierarchy in response to a miss in the first cache.

59. The method of claim 58, wherein the act (A) comprises an act of dynamically modifying the stacked hierarchy by at least one of adding and removing a cache from the stacked hierarchy.

60. The method of claim 59, wherein the stacked hierarchy is arranged so that data replaced out of the first cache is automatically demoted to the second cache, and data that hits in the second cache is automatically promoted to the first cache by the cache hierarchy.

61. The method of claim 55, wherein the plurality of software caches comprises a number of software caches, and wherein the act (A) comprises an act of modifying the number of software caches in the cache arrangement.

62. The method of claim 55, wherein the act (A) comprises an act of dynamically altering a storage medium used to implement at least one of the plurality of caches.

63. The method of claim 58, wherein the stacked hierarchy is arranged such that addresses are compared first against the first cache, and are compared against the second cache only in response to a miss in the first cache.

64. The method of claim 55, wherein the act (A) comprises an act of dynamically modifying a size of at least one of the plurality of caches.

65. The method of claim 55, wherein the act (A) comprises an act of dynamically modifying a hashing algorithm of at least one of the plurality of caches.

66. The method of claim 55, wherein the act (A) comprises an act of dynamically removing at least one of the plurality of caches from the cache arrangement.

67. The method of claim 55, wherein the act (A) comprises an act of dynamically adding at least one cache to the cache arrangement.

68. The method of claim 55, wherein the act (A) comprises an act of dynamically adding to the cache arrangement at least one cache that employs a different hashing function than the other of the plurality of caches in the cache arrangement.

69. A computer for use in a computer system, the computer comprising:

a cache arrangement comprising a plurality of caches that are interrelated to form the cache arrangement; and

at least one controller capable of dynamically reconfiguring the cache arrangement without reconfiguring an application, executing on the computer system, that accesses the cache arrangement.

70. The computer of claim 69, further comprising at least one processor and at least one memory to store software to be executed on the at least one processor, wherein the cache arrangement and the at least one controller each is implemented by software executing on the at least one processor.

71. The computer of claim 69, wherein at least one of the plurality of caches is a set associative cache comprising a number of cache groups each including a number of cache sets, and wherein the at least one controller is capable of dynamically reconfiguring the at least one of the plurality of caches by modifying at least one of the number of cache groups and the number of cache sets.

72. The computer of claim 69, wherein the plurality of caches are arranged in a stacked hierarchy, such that addresses are compared first against a first cache in the stacked hierarchy, and are compared against a second cache in the stacked hierarchy in response to a miss in the first cache.

73. The computer of claim 72, wherein the at least one controller is capable of dynamically modifying the stacked hierarchy by at least one of adding and removing a cache from the stacked hierarchy.

74. The computer of claim 73, wherein the stacked hierarchy is arranged so that data replaced out of the first cache is automatically demoted to the second cache, and data that hits in the second cache is automatically promoted to the first cache by the cache hierarchy.

75. The computer of claim 69, wherein the plurality of software caches comprises a number of software caches, and wherein the at least one controller is capable of modifying the number of software caches in the cache arrangement.

76. The computer of claim 69, wherein the at least one controller is capable of dynamically altering a storage medium used to implement at least one of the plurality of caches.

77. The computer of claim 72, wherein the stacked hierarchy is arranged such that addresses are compared first against the first cache, and are compared against the second cache only in response to a miss in the first cache.

78. The computer of claim 69, wherein the at least one controller is capable of dynamically modifying a size of at least one of the plurality of caches.

79. The computer of claim 69, wherein the at least one controller is capable of dynamically modifying a hashing algorithm of at least one of the plurality of caches.

80. The computer of claim 69, wherein the at least one controller is capable of dynamically removing at least one of the plurality of caches from the cache arrangement.

81. The computer of claim 69, wherein the at least one controller is capable of dynamically adding at least one cache to the cache arrangement.

82. The computer of claim 69, wherein the at least one controller is capable of dynamically adding to the cache arrangement at least one cache that employs a different hashing function than the other of the plurality of caches in the cache arrangement.

83. A computer readable medium encoded with a program for execution on a computer system having a cache arrangement, the cache arrangement having a plurality of caches that are interrelated to form the cache arrangement, wherein the program, when executed, performs a method for managing the cache arrangement, the method comprising an act of:

(A) dynamically reconfiguring the cache arrangement without reconfiguring an application, executing on the computer system, that accesses the cache arrangement.

84. The computer readable medium of claim 83, wherein the plurality of caches comprises a plurality of software caches.

85. The computer readable medium of claim 83, wherein at least one of the plurality of caches is a set associative cache comprising a number of cache groups each including a number of cache sets, and wherein the act (A) comprises an act of dynamically reconfiguring the at least one of the plurality of caches by modifying at least one of the number of cache groups and the number of cache sets.

86. The computer readable medium of claim 83, wherein the plurality of caches are arranged in a stacked hierarchy, such that addresses are compared first against a first cache in the stacked hierarchy, and are compared against a second cache in the stacked hierarchy in response to a miss in the first cache.

87. The computer readable medium of claim 86, wherein the act (A) comprises an act of dynamically modifying the stacked hierarchy by at least one of adding and removing a cache from the stacked hierarchy.

88. The computer readable medium of claim 87, wherein the stacked hierarchy is arranged so that data replaced out of the first cache is automatically demoted to the second cache, and data that hits in the second cache is automatically promoted to the first cache by the cache hierarchy.

89. The computer readable medium of claim 83, wherein the plurality of software caches comprises a number of software caches, and wherein the act (A) comprises an act of modifying the number of software caches in the cache arrangement.

90. The computer readable medium of claim 83, wherein the act (A) comprises an act of dynamically altering a storage medium used to implement at least one of the plurality of caches.

91. The computer readable medium of claim 86, wherein the stacked hierarchy is arranged such that addresses are compared first against the first cache, and are compared against the second cache only in response to a miss in the first cache.

92. The computer readable medium of claim 83, wherein the act (A) comprises an act of dynamically modifying a size of at least one of the plurality of caches.

93. The computer readable medium of claim 83, wherein the act (A) comprises an act of dynamically modifying a hashing algorithm of at least one of the plurality of caches.

94. The computer readable medium of claim 83, wherein the act (A) comprises an act of dynamically removing at least one of the plurality of caches from the cache arrangement.

95. The computer readable medium of claim 83, wherein the act (A) comprises an act of dynamically adding at least one cache to the cache arrangement.

96. The computer readable medium of claim 83, wherein the act (A) comprises an act of dynamically adding to the cache arrangement at least one cache that employs a different hashing function than the other of the plurality of caches in the cache arrangement.